Reinforcement learning under circumstances beyond its control

Chris Gaskett[†]

CyberHuman Project, Department 3, Human Information Science Laboratories,

ATR International, Kyoto, Japan

cgaskett@atr.co.jp, http://www.his.atr.co.jp/∼cgaskett/

Abstract

Decision theory addresses the task of choosing an action; it provides robust decision-making criteria that support decision-making under conditions of uncertainty or risk. Decision theory has been applied to produce reinforcement learning algorithms that manage uncertainty in state-transitions. However, performance when there is uncertainty regarding the selection of future actions must also be considered, since reinforcement learning tasks are multiple-step decision problems. This work proposes β-pessimistic $\mathcal{Q}$-learning—a reinforcement learning algorithm that does not assume complete control.

1 Introduction

In reinforcement learning tasks the learning system must discover by trial-and-error which actions are most valuable in particular states [1]. Generally, no dynamic model of the environment is available a priori; the performance of learning system improves incrementally through interaction with the environment.

In reinforcement learning nomenclature the *state*, x, is a representation of the current situation in the learning system's environment. The *action*, u, is an output from the learning system that can influence its environment. The learning system's choice of actions in response to states is called its *policy*. Actions should be chosen with the future in mind, rather than just the immediate payoff. Thus, reinforcement learning tasks are multiple-step decision problems. Evaluative feedback is provided to the learning system in the form of a scalar *reward* signal, r, that may be delayed. The reward signal is defined in relation to the task to be achieved; reward is given when the system is successfully achieving the task.

Reinforcement learning has traditionally emphasised the criterion of expected value, defining the optimal action as the action that will result in the highest expected (mean)

[†]Parts of this research were performed at the Robotic Systems Laboratory, Department of Systems Engineering, RSISE, The Australian National University, Canberra, ACT 0200 Australia, http://www.syseng.anu.edu.au/rsl/

reward under the assumption that actions with the highest expected value will be executed thereafter. Thus, the design of the learning algorithm assumes that it is the sole decision-maker in the system.

Decision theory offers alternative decision-making criteria that result in more robust behaviour in situations with risk, disturbances, or unexpected events [2]. Several risk-averse reinforcement learning algorithms have already been developed based on ideas from decision theory. This work suggests expansion of these ideas, based on the idea that a robust decision policy should not expect to be in complete control of all future actions.

2 Decision Theory, Uncertainty, and Risk

Decision theory addresses the problem of choosing an action from a set of actions [2]. Each action produces a resulting *value*. Under *strict uncertainty*, we have no idea which value will follow, whereas under *risk*, the probability distribution of the result values is known and a more informed decision can be made. Action choice under strict uncertainty could be based on various criteria, for example:

- the action with the highest potential value (extreme optimism);
- the action with the highest mean value (the *Laplace* criterion); or
- the action with the highest minimum value (the *Wald* or *minimax* criterion).

Various reinforcement learning and dynamic programming algorithms have been proposed for dealing with *nondeterministic state transitions*, in which actions do not produce a completely predictable effect on the state. Several methods use the minimax criterion, choosing actions under the assumption that the worst possible state transition will occur [3–6], or similarly, that the worst possible disturbance to the state will occur [7]. Other, *risk-sensitive*, dynamic programming methods operate under risk, rather than uncertainty, and use knowledge of the distribution of state transitions to produce a risk-averse policy [6]. Neuneier and Mihatsch's [8] model-free method achieved the same goal by emphasising unexpected negative results and de-emphasising unexpected positive results. Geibel [9] defined risk differently, seeing it as the probability of reaching a fatal state, and defined an algorithm that compromised between achieving the task and keeping the risk of fatality low.

Beyond the uncertainty in state-transitions, reinforcement learning systems must deal with another uncertainty problem: uncertainty surrounding *the choice of future actions* after the current action has been executed [1, 10]. One source of unpredictable actions is exploration: In order to find a good policy, a learning system must try various actions, under various states, and evaluate the results. Interaction between different behavioural modules can also lead to unpredictable actions. For example, if one learned behaviour is to avoid obstacles, and another is to move toward a goal, a compromise action may not follow the policy of either behaviour. The action could be a mixture of the two behaviour's actions, or behaviours could become dominant depending on the state. Other unexpected actions could be produced, for example, by a safety monitor that vetoes some of the actions suggested by the learning algorithm.

Consequently, a decision policy that relies on the future execution of an exact series of actions is fragile. A robust decision policy should be somewhat conservative, and favour states where a spurious action or two will not cause serious harm. Even if no uncertainty is expected, compensating for some uncertainty is a reasonable method of improving robustness.

3 Cliff-Walking: An Example of Decisions Under Uncertainty

The cliff-walking task is used to compare the characteristics of algorithms under uncertainty. The task, proposed by Sutton and Barto [1], is to walk across a plateau at the top of a dangerous cliff in order to reach a goal position. The situation is shown in Figure 1. The player starts above the point marked S. Each step has a cost (negative reward) of 1. Falling off the cliff has a cost of 100 and ends the attempt. Stepping onto the goal, G, has a cost of zero and ends the task. Other attempts to move off the play area do not move the player.

The task would be trivial to solve except that under *nondeterministic action-selection* 10% of the actions are chosen randomly. Early in learning this is an advantage: it provides exploration so that the agent can learn the values of various actions. Later, it is a hindrance that causes falls from the cliff[1].

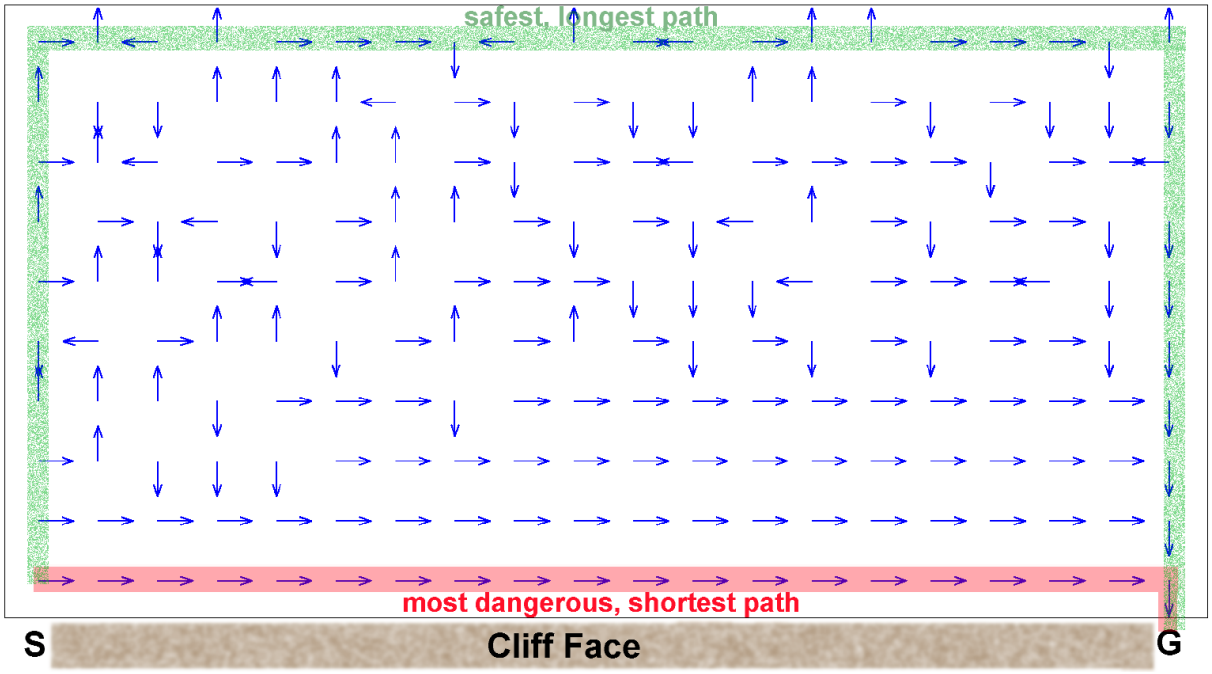

Figure 1: The cliff-walking problem. $\mathcal{Q}$-learning under nondeterministic action-selection found a risky solution.

[1]To make the task fairer, and a little less gruesome, the randomness never causes the player to fall off the cliff from the start square. Otherwise $\frac{1}{10}$ of $\frac{1}{4}$ of the time the player would fall off the cliff immediately, regardless of the player's policy.

4 Algorithms

The following sections describe $\mathcal{Q}$-learning and several variations of $\mathcal{Q}$-learning, comparing their behaviours using the cliff-walking task. We consider two possible learning conditions:

- nondeterministic action-selection: 10% of actions are chosen randomly, rather than following the choice of the learning system; and
- nondeterministic state-transitions: 10% of actions cause a state transition in a random direction.

Figures 1 to 7 demonstrate the learned decision policies after 10,000 attempts to cross the cliff. In interpreting the pictures it is not useful to assign meaning to every single arrow (the decision policy in every state), since some states have been visited rarely and the action-values are still in flux. Nevertheless, the overall trend in the policies is clear. The graphs in Figures 8 and 9 show the performance of the algorithms over time, and Table 1 summarises the policies developed by the learning algorithms.

4.1 $\mathcal{Q}$-Learning

$\mathcal{Q}$-learning is an algorithm for solving reinforcement learning tasks [11]. $\mathcal{Q}$-learning stores the *expected value*, $\mathcal{Q}(x, u)$, of performing each action in each state, assuming that the actions with the highest expected values will be performed thereafter.

The action-values are updated according to:

$$\mathcal{Q}(x_t, u_t) \xleftarrow{\alpha} r(x_t, u_t, x_{t+1}) + \gamma \max_{u_{t+1}} \mathcal{Q}(x_{t+1}, u_{t+1}) \tag{1}$$

where α is the learning rate (or step size), between 0 and 1, that controls convergence; and γ is the discount factor, between 0 and 1, that makes rewards that are earned later exponentially less valuable.

$\mathcal{Q}$-learning uses the Laplace criterion, the expected value, to compare the value of states. Thus, it is risk-neutral with respect to the effects of nondeterministic state-transitions.

With regards to the effect of nondeterministic action-selection, $\mathcal{Q}$-learning is *optimistic*: it assumes that actions with the highest known value will be executed thereafter. Figure 1 illustrates $\mathcal{Q}$-learning's optimism[2]. It learnt to walk along the edge of the cliff, under the assumption that no unexpected actions will occur. Consequently, the player fell often.

4.2 Compensating for Exploration

John [10] addressed a subproblem of nondeterministic action-selection—coping with exploration—by including knowledge of the exploration policy within the learning update. This is applicable only when the nature of the variations to the decision policy are known in advance.

[2]The parameter settings were $\alpha = 0.5$, and $\gamma = 1.0$. Action-values were stored in a table with each combination of state and action filling a cell.

4.3 Sarsa

The Sarsa algorithm addresses the problem of nondeterministic action-selection by modifying the $\mathcal{Q}$-learning update to use the value of the *executed* action [1, 12, 13]. Sutton and Barto [1] introduced the cliff-walking problem to demonstrate Sarsa's ability to deal with nondeterministic action-selection. The Sarsa-action-values are updated as follows:

$$\mathcal{Q}_{Sarsa}(x_t, u_t) \xleftarrow{\alpha} r(x_t, u_t, x_{t+1}) + \gamma \mathcal{Q}_{Sarsa}(x_{t+1}, u_{t+1}) \tag{2}$$

Sarsa stands for **s**tate, **a**ction, **r**eward, next **s**tate, and next **a**ction. The Sarsa-action-values are the sum of the reward for performing the current action and the expected rewards for continuing to follow the policy that has been followed in the past. If actions are always executed as chosen by the learning system then the Sarsa algorithm is equivalent to standard $\mathcal{Q}$-learning.

Figure 2 shows the solution found by the Sarsa algorithm. Sarsa took into account that the favoured action may not always be performed and took a safer path by stepping a little away from the cliff edge.

A limitation of Sarsa is that it cannot perform *off-policy learning* [1]. The decision policy learnt by Sarsa is heavily dependent on the control policy it observes. Sarsa can learn little from observing the effects of strings of random actions, whereas standard $\mathcal{Q}$-learning can learn a policy that is optimal with respect to the reward function. In our experiment, Sarsa learnt a safe path across the cliff because some actions were chosen randomly. If nondeterministic action-selection ceases then Sarsa reverts to the same dangerous path chosen by standard $\mathcal{Q}$-learning. Sarsa has no facility for manual adjustment between optimism and pessimism; its behaviour changes based on the control policy it observes.

4.4 $\widehat{\mathcal{Q}}$-Learning—The Minimax Criterion

Heger [5] developed a risk-averse variation of $\mathcal{Q}$-learning called $\widehat{\mathcal{Q}}$-learning ("q-hat"), using the minimax criterion. The algorithm addresses the problem of nondeterministic state-transitions, rather than nondeterministic action-selection. However, it is discussed here as it demonstrates the characteristics of minimax methods.

In the $\widehat{\mathcal{Q}}$ approach, the action-values encode the *worst-case* value of performing an action, assuming that actions with the highest known worst-case value will be executed thereafter. It deals with the nondeterministic state-transition problem by assuming that the world is adversarial—that the worst possible *state transitions* will occur. This pessimism leads to highly conservative policies. The $\widehat{\mathcal{Q}}$-action-value effectively gives a lower bound on value, or an upper bound on total cost. It is a minimax approach because the policy prefers actions which minimise the maximum possible total cost.

The $\widehat{\mathcal{Q}}$-action-values are updated as follows[3]:

$$\widehat{\mathcal{Q}}(x_t, u_t) \longleftarrow \min\left[\widehat{\mathcal{Q}}(x_t, u_t) \ , \ r(x_t, u_t, x_{t+1}) + \gamma \max_{u_{t+1}} \widehat{\mathcal{Q}}(x_{t+1})\right] \tag{3}$$

[3]The formulation is as given by Koenig and Simmons [14]. It is equivalent to the original formulation, in terms of costs, as given by Heger [5].

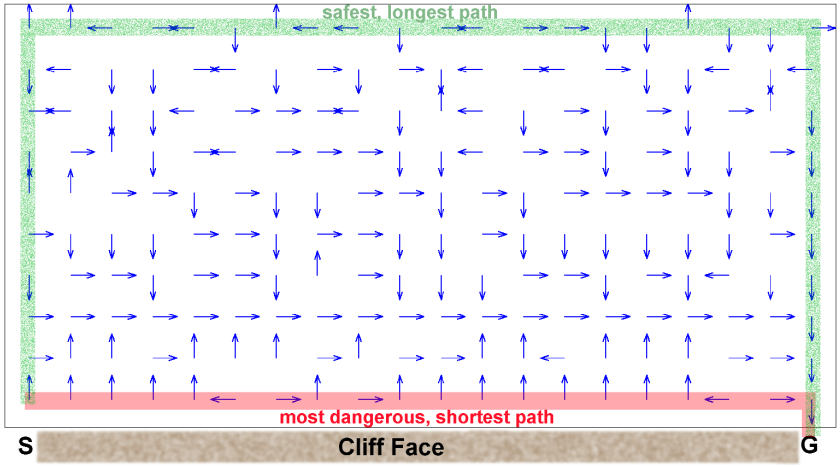

Figure 2: Sarsa under nondeterministic action-selection found a safe solution.

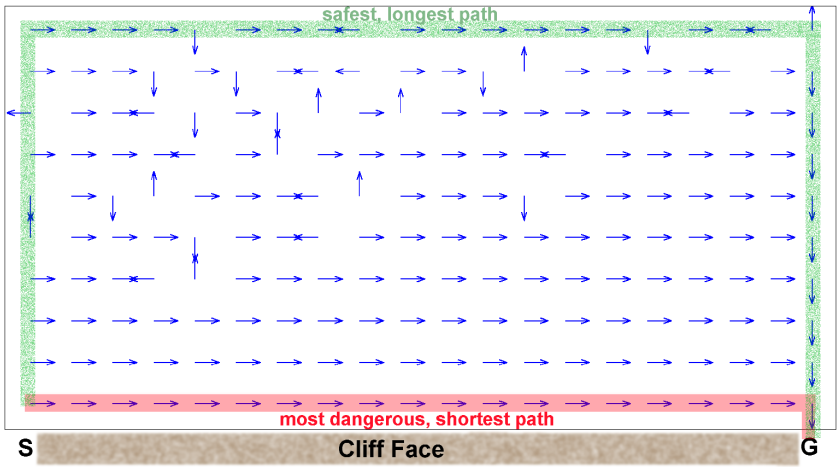

Figure 3: Heger's $\widehat{\mathcal{Q}}$ algorithm under nondeterministic action-selection found a risky solution.

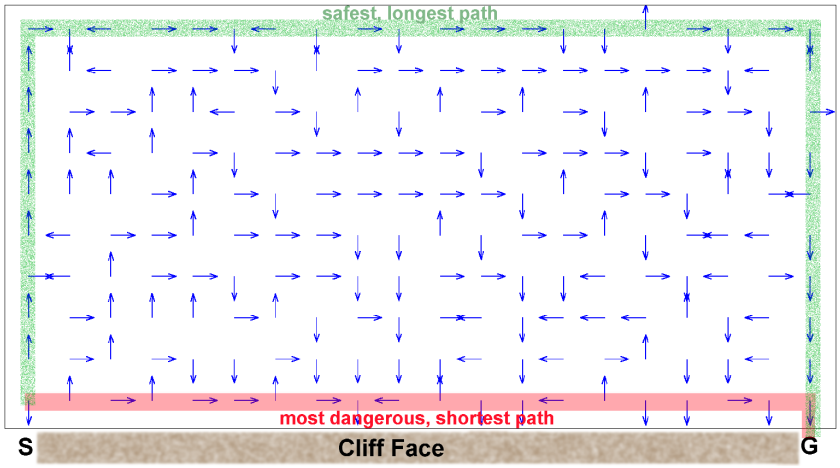

Figure 4: Heger's $\widehat{\mathcal{Q}}$-learning under nondeterministic state-transitions shows its extreme pessimism. The player jumps off the cliff from the *start* square to avoid the higher total cost of falling later.

The minimum operator in Equation (3) is a cause for concern. Since the action-values can only move downward in value they must be initialised optimistically. If the rules of the world change the action-values cannot rise to fit the new situation.

Under nondeterministic action-selection $\widehat{\mathcal{Q}}$-learning is similar to $\mathcal{Q}$-learning. Accordingly, $\widehat{\mathcal{Q}}$-learning chose the same dangerous path as $\mathcal{Q}$-learning, as shown in Figure 3.

$\widehat{\mathcal{Q}}$-learning demonstrates its extreme pessimism under nondeterministic state transitions. Figure 4 shows that the player jumps off the cliff from the *start* square, at a cost of 100, to avoid the possibility of taking a few steps, falling accidentally and incurring slightly higher total cost[4]. The pessimism of minimax methods is well-known, it makes them appropriate for adversarial games, and often inappropriate for other problems. For example, an investment manager following the minimax approach would never make investments [8].

4.5 β-Pessimistic $\mathcal{Q}$-Learning

This work proposes a new extension to $\mathcal{Q}$-learning, β-pessimistic $\mathcal{Q}$-learning, that is appropriate for tasks under nondeterministic action-selection. β-pessimistic learning compromises between the extreme optimism of standard $\mathcal{Q}$-learning and the extreme pessimism of minimax approaches.

The approach is based on the Hurwicz α-criterion, which uses a mixture between the maximum and the minimum controlled by a pessimism parameter [2]. The parameter, β, acts in a similar way: the β-pessimistic action-values represent the expected value of performing an action followed by actions which are highest valued with probability $1-\beta$, or lowest valued with probability β. Thus when β is zero we have standard $\mathcal{Q}$-learning, and when β is one we have a minimax algorithm.

The β-pessimistic update is as follows:

$$\begin{aligned}\mathcal{Q}_\beta(x_t,u_t) \xleftarrow{\alpha} & r(x_t,u_t,x_{t+1}) \\ & + \gamma\left[(1-\beta)\max_{u_{t+1}}\mathcal{Q}_\beta(x_{t+1},u_{t+1}) + \beta\min_{u_{t+1}}\mathcal{Q}_\beta(x_{t+1},u_{t+1})\right],\end{aligned} \tag{4}$$

clearly showing the mixture of maximum and minimum action-values, or rearranged to form:

$$\begin{aligned}\mathcal{Q}_\beta(x_t,u_t) \xleftarrow{\alpha} & r(x_t,u_t,x_{t+1}) \\ & + \gamma\left[\max_{u_{t+1}}\mathcal{Q}_\beta(x_{t+1},u_{t+1}) - \beta\left(\max_{u_{t+1}}\mathcal{Q}_\beta(x_{t+1},u_{t+1}) - \min_{u_{t+1}}\mathcal{Q}_\beta(x_{t+1},u_{t+1})\right)\right],\end{aligned} \tag{5}$$

which shows a penalty for variation between maximum and minimum action-values; thus demonstrating the idea that safe areas in state space aren't strongly affected by which action is chosen.

[4]Dr. Jochen Heinzmann noted that it may be unrealistic to give a higher total cost to taking a few steps and falling off the cliff than to falling off the cliff immediately. This could be termed "Heinzmann's dead-is-dead conjecture". Separating fatal from normal states may be a more appropriate framework for these types of problems [9].

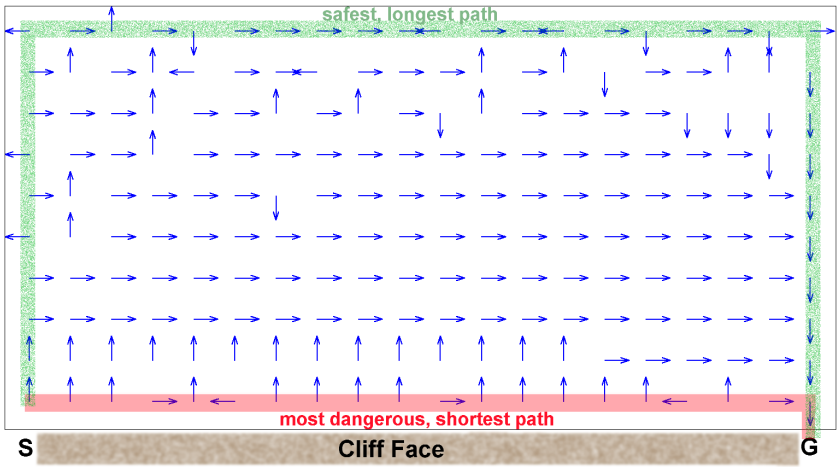

Figure 5: The β-pessimistic method with pessimism set to 0.1 under nondeterministic action-selection took a safe path by taking a few steps away from the cliff.

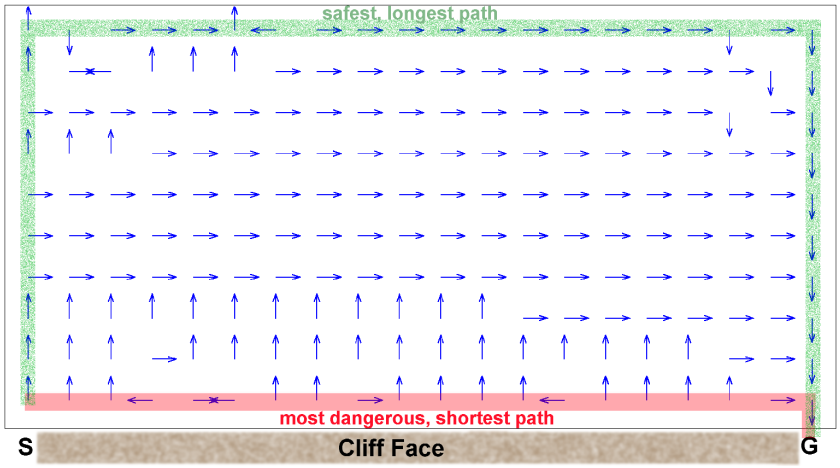

Figure 6: The β-pessimistic method with pessimism set to 0.2 under nondeterministic action-selection took a safer path by taking several steps away from the cliff.

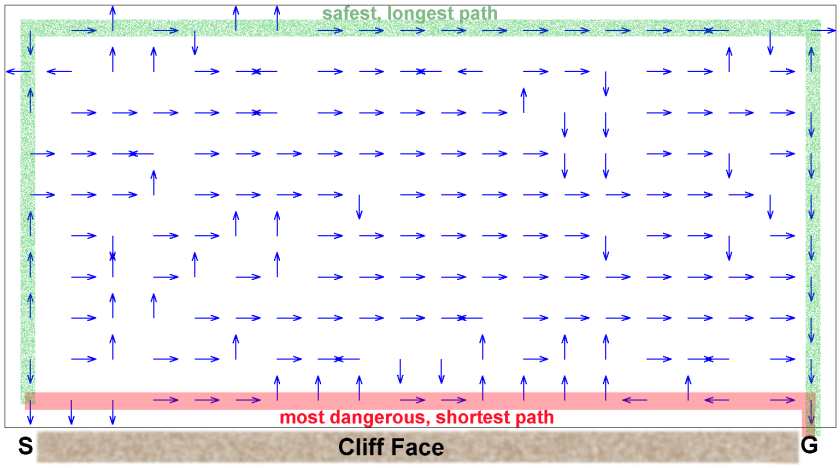

Figure 7: The β-pessimistic method with pessimism set to 0.5 under nondeterministic action-selection jumps off the cliff from the start square.

To demonstrate the algorithm and the effect of adjusting β, three diagrams are shown in Figures 5, 6, and 7, for β equal to 0.1, 0.2, and 0.5 respectively. When $\beta = 0.1$, the player took a safe path, stepping away from the cliff because the learning algorithm assumed that the worst possible action would be selected 10% of the time. When $\beta = 0.2$, the player was more pessimistic, and took a safer path.

When β was increased to 0.5 (Figure 7), the player elected to jump from the cliff from the first square. Under the assumption that 50% of future actions would be the worst possible it followed the same logic as the $\widehat{\mathcal{Q}}$ algorithm (Figure 4). A difference from the $\widehat{\mathcal{Q}}$ algorithm is that if the player managed to get close to the goal position it attempted to reach the goal.

There was a subtle difference between the policies found by the β-pessimistic algorithms and the Sarsa algorithm. Under the β-pessimistic policy, if the player was several steps away from the cliff the player walked straight to the right until they reached the extreme right hand side, then walk straight down toward the goal (Figure 5). Under Sarsa, the player walked diagonally down and to the right instead of all the way to the right edge, a single row two steps from the cliff was preferred (Figure 2). A similar effect can be observed on the left-hand side of the diagrams. Sarsa's stair-like diagonal path has the same length as the right angle path, yet keeps the player closer to the cliff edge longer and, consequently, is more dangerous.

5 Discussion

The graphs in Figures 8 and 9 show the performance of the algorithms. The results show the total cost of attempts 1 to 5000, averaged over 1000 simulation runs of each algorithm. The policies developed by the different learning methods are summarised in Table 1. β-pessimistic learning achieved the best result for the nondeterministic action-selection case. The graphs show that the β-pessimistic algorithm is suitable for tasks involving both nondeterministic state-transitions and action-selection.

When $\beta = 0.5$ the β-pessimistic algorithm was too pessimistic, as is $\widehat{\mathcal{Q}}$-learning under nondeterministic state-transitions. However, the graph shows that the average total cost when $\beta = 0.5$ is slightly better than 100, whereas for $\widehat{\mathcal{Q}}$-learning it is worse than 100. Under nondeterministic state transitions, some actions do not produce movement in the expected direction; therefore, the player does not always manage to jump off the cliff from the start square. When $\beta = 0.5$, a player that does not jump off the cliff may reach the goal square, thus lowering the average total cost. However, $\widehat{\mathcal{Q}}$-learning did not converge to a consistent strategy after 10,000 attempts. As shown in Figure 4, players that did not jump off the cliff immediately took a long, rambling path, incurring high total cost.

The results for $\mathcal{Q}$-learning and Sarsa for nondeterministic action-selection accord with the results given by Sutton and Barto [1], with Sarsa having better performance. Under nondeterministic state-transitions, $\mathcal{Q}$-learning and Sarsa are equivalent. Unlike Sarsa, β-pessimistic learning and $\mathcal{Q}$-learning are capable of off-policy learning. Therefore, they can learn by observing the actions of other players. β-pessimistic learning will maintain a conservative approach even when it is in complete control and exploration has ceased.

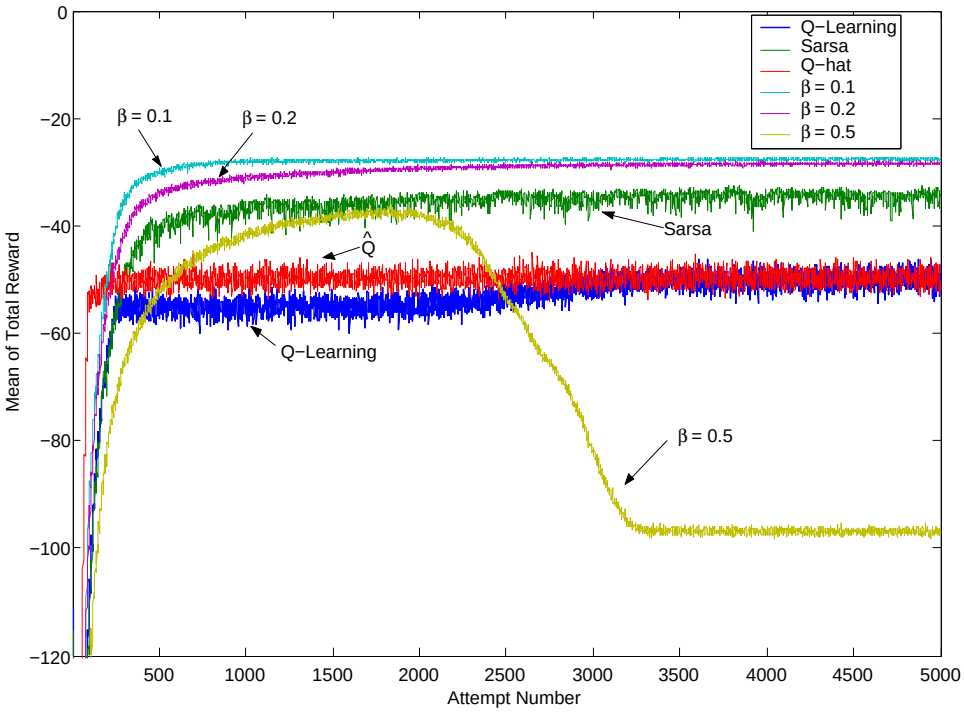

Figure 8: Performance of various algorithms solving the cliff-walking task under nondeterministic action-selection. Results below -120 are not shown.

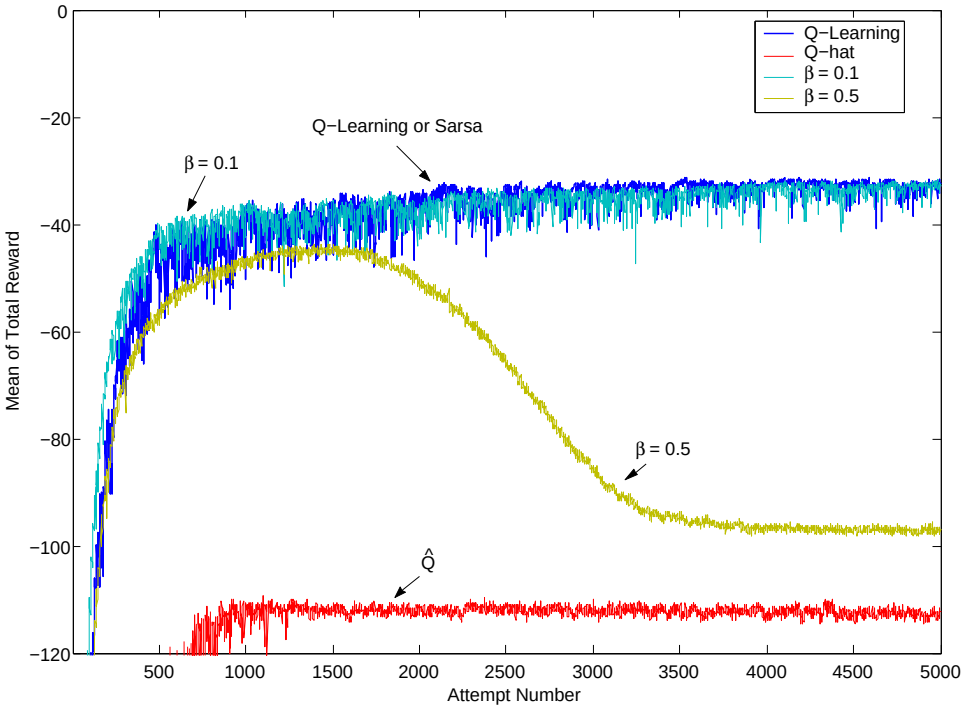

Figure 9: Performance of various algorithms solving the cliff-walking task under nondeterministic state-transitions

Table 1: Summary of the learned policies for the cliff-walking problem

Method	Learned Policy	Figure
	nondeterministic action-selection	
$\mathcal{Q}$	unsafe; walks along the very edge of the cliff (falls often)	1
Sarsa	safe; walks a little away from the edge then to the goal, but only as long as exploratory actions are performed	2
$\widehat{\mathcal{Q}}$	unsafe; similar to $\mathcal{Q}$, but explores more	3
$\beta = 0.1$	safe; walks a little away from the edge then to the goal	5
$\beta = 0.2$	safer; walks further away from the edge then to the goal	6
$\beta = 0.5$	too pessimistic; if far from the goal, jumps off the cliff to avoid falling off later	7
	nondeterministic state-transitions	
$\mathcal{Q}$	safe; walks a little away from the edge	-
Sarsa	safe; same as $\mathcal{Q}$	-
$\widehat{\mathcal{Q}}$	too pessimistic; jumps off the cliff to avoid falling off later	4
$\beta = 0.1$	safe; similar to $\mathcal{Q}$ but walks farther from the edge	-

β-pessimistic learning converges under the same conditions as $\mathcal{Q}$-learning and $\widehat{\mathcal{Q}}$-learning, based on the convergence proofs by Littman and Szepesvári [15], and Heger [16], since the operation of taking the fixed weighted sum of the maximum and the minimum is a non-expansion. It would be simple to devise other converging update rules, using the same idea that a robust solution should tolerate occasional spurious actions. For example, a weighted sum of the highest valued action, and the average value of other actions could be used. Such a scheme might be reasonable for problems with discrete actions; it would not, however, be suitable for systems that deal with continuously variable actions, in which the action-values are approximated [17]. Under most approximation schemes it would be difficult to calculate the average of the action-values. In contrast, β-pessimistic learning only requires the maximum and the minimum. $\mathcal{Q}$-learning already requires finding the maximum, and finding the minimum is an equivalent task.

6 Conclusion

β-pessimistic learning is a promising algorithm which can find policies which are robust to the effects of nondeterministic action-selection. Unlike Sarsa, it is capable of off-policy learning. It is also computationally feasible, even for learning problems with continuous actions. The factor β allows adjustment between optimism and pessimism. Beyond the particular algorithm, considering nondeterministic action-selection leads to a robust learning system that seeks decision policies tolerant of actions beyond its control.

Acknowledgements

Thank you to Professor Alexander Zelinsky, Dr. Gordon Cheng, Dr. Jonathan Baxter, Dr. Jun Morimoto, and Dr. Jochen Heinzmann for their comments and encouragement.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Bradford Books, MIT, 1998.

[2] S. French, *Decision Theory: an introduction to the mathematics of rationality*, Ellis Horwood, 1986.

[3] D. P. Bertsekas, *Control of Uncertain Systems with a Set-Membership Description of the Uncertainty*, Ph.D. thesis, Dept. of Electrical Engineering, MIT, 1971.

[4] M. L. Littman, *Algorithms for Sequential Decision Making*, Ph.D. thesis, Brown University, 1996.

[5] M. Heger, "Consideration of risk in reinforcement learning," in *Proc. of the 11th International Machine Learning Conference, ML'94*, 1994.

[6] S. Coraluppi and S. Marcus, "Risk-sensitive and minimax control of discrete-time, finite-state markov decision processes," *Automatica*, vol. 35:pp. 301–309, 1999.

[7] J. Morimoto and K. Doya, "Robust reinforcement learning," in *Advances in Neural Information Processing Systems 13*, pp. 1061–1067, MIT Press, 2001.

[8] R. Neuneier and O. Mihatsch, "Risk sensitive reinforcement learning," in *Advances in Neural Information Processing Systems 11*, pp. 1031–1037, MIT Press, 1999.

[9] P. Geibel, "Reinforcement learning with bounded risk," in *Proc. of the 18th International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, 2001.

[10] G. H. John, "When the best move isn't optimal: Q-learning with exploration," in *Proc. of the Tenth National Conference on Artificial Intelligence (student abstract)*, Menlo Park, CA, 1994.

[11] C. J. C. H. Watkins and P. Dayan, "Technical note: Q learning," *Machine Learning*, vol. 8(3/4):pp. 279–292, 1992.

[12] G. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," Tech. Rep. CUED/F-INFEG/TR66, Cambridge University, 1994.

[13] G. A. Rummery, *Problem solving with reinforcement learning*, Ph.D. thesis, Cambridge University, 1995.

[14] S. Koenig and R. G. Simmons, "The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms," *Machine Learning*, vol. 22, 1996.

[15] M. L. Littman and C. Szepesvári, "A generalized reinforcement-learning model: Convergence and applications," in *Proc. of the 13th International Conference on Machine Learning*, 1996.

[16] M. Heger, "The loss from imperfect value functions in expectation-based and minimax-based tasks," *Machine Learning*, vol. 22, 1996.

[17] C. Gaskett, D. Wettergreen, and A. Zelinsky, "Q-learning in continuous state and action spaces," in *Proc. of the 12th Australian Joint Conference on Artificial Intelligence*, Sydney, Australia, 1999.